COMPUTER PROGRAMMING

ASSIGNMENT # 1

OCTOBER 25, 2023

TALHA NAVEED AND MUHAMMAD SIAWISH
01-131232-089 AND 01-131232-025

ASSIGNMENT 1: PROBLEM SOLVING (CL 02)

ALGORITHM # 1

Question 1: Finding the Shortest Path

Imagine you are developing a GPS navigation system. You are given a map with various locations and the roads connecting them. Your task is to write an algorithm to find the shortest path from one location to another. You can assume that you have a list of locations and the distance between each pair of locations. Your algorithm should output the shortest path and the total distance.

INPUT: A WITH LIST OF LOCATIONS AND DISTANCES BETWEEN THOSE LOCATIONS.

OUTPUT: SHORTEST PATH FROM CURRENT LOCATION TO THE DESTINATION LOCATION AND TOTAL DISTANCE OF THE SHORTEST PATH.

SOLUTION:

STEP 1: START.

STEP 2: CREATE A LIST THAT WILL STORE DISTANCES FROM CURRENT LOCATION TO ALL THE OTHER LOCATIONS.

STEP 3: INITIALIZE A LIST THAT STORES THE DISTANCES OF THE ROADS CONNECTING THE LOCATIONS TO ONE ANOTHER.

STEP 4: ASK THE USER TO INPUT HIS CURRENT LOCATION.

STEP 5: ASK THE USER TO INPUT THE DESTINATION LOCATION.

STEP 6: INITIALIZE THE CURRENT LOCATION TO ZERO (0) AND ALL THE OTHER LOCATIONS WITH A VERY LARGE NUMBER.

STEP 7: CREATE A LIST TO STORE THE UNVISITED LOCATIONS FOR EACH LOCATION.

STEP 8: INITIALIZE ALL THE PREVIOUS LOCATIONS TO NULL.

STEP 9: FIND THE LOCATION WITH THE SMALLEST DISTANCE FROM THE CURRENT LOCATION AMONG THE UNVISITED LOCATIONS.

STEP 10: NOW MARK THIS LOCATION AS VISITED.

STEP 11: NOW SELECT THIS VISITED LOCATION AND FIND THE DISTANCE TO ALL OF IT'S NEIGHBOURS.

STEP 12: IF A SHORTER PATH IS FOUND THEN UPDATE THE DISTANCES.

STEP 13: REPEAT STEP 9 TO STEP 12 UNTIL THE SHORTEST PATH TO ALL THE LOCATIONS IS FOUND.

STEP 14: NOW USE THE PREVIOUS LOCATIONS LIST TO BACKTRACK FROM DESTINANTION LOCATION TO CURRENT LOCATION.

STEP 15: OUTPUT THE SHORTEST PATH.

STEP 16: ADD THE DISTANCES BETWEEN EACH PAIR OF CONSECUTIVE LOCATIONS IN THE PATH FOUND IN STEP 14 TO FIND THE TOTAL DISTANCE OF THE SHORTEST PATH.

STEP 17: OUTPUT THE TOTAL DISTANCE.

STEP 18: STOP.

ALGORITHM # 2

Question 2: Sorting a List of Numbers

You are working on a project where you need to sort a list of numbers in ascending order. Design an algorithm to efficiently sort a list of integers. You should consider various sorting algorithms, evaluate their time complexity, and choose the most suitable one for the task.

INPUT: A LIST OF INTEGERS.

OUTPUT: SORTED LIST IN ASCENDING ORDER.

SOLUTION:

STEP 1: START.

STEP 2: INITIALIZE A LIST OF NUMBERS.

STEP 3: CHOOSE A PIVOT ELEMENT FROM THE LIST OF NUMBERS. SELECT ANY RANDOM ELEMENT.

STEP 4: REARRANGE THE LIST INTO SUB-LIST1 AND SUB-LIST2 BY COMPARING ALL THE ELEMENTS OF THE LIST WITH THE PIVOT ELEMENT.

STEP 5: IF AN ELEMENT IS SMALLER THEN THE PIVOT ELEMENT IT WILL COME BEFORE THE PIVOT ELEMENT IN SUB-LIST 1 .

STEP 6: IF AN ELEMENT IS GREATER THEN THE PIVOT ELEMENT IT WILL COME AFTER THE PIVOT ELEMENT IN SUB-LIST2.

STEP 7: RECURSION:

- REPEAT STEP 2 TO STEP 6 FOR SUB-LISTS UNTIL THE SUBLISTS REACHES A BASE CASE. (A BASE CASE IS WHEN A SUBLIST ONLY HAS ONE OR ZERO ELEMENTS, WHICH IS ALREADY SORTED).

STEP 8: RETURN THE SORTED LIST.

STEP 9: STOP

ALGORITHM # 3

Question 3: Calculating Fibonacci Numbers

The Fibonacci sequence is a series of numbers where each number is the sum of the two preceding ones (e.g., 0, 1, 1, 2, 3, 5, 8, 13, ...). Write an algorithm to calculate the nth Fibonacci number. Your algorithm should be efficient and capable of handling large values of n.

INPUT: NTH FIBONACCI NUMBER.

OUTPUT: VALUE OF THE NTH FIBONACCI NUMBER.

SOLUTION:

STEP 1: START.

STEP 2: **INITIALIZE TWO long integer VARIABLES A=0 AND B=1.**

STEP 3: **DECLARE A VARIABLE N FOR TAKING INPUT FROM THE USER.**

STEP 4: **INPUT N FROM THE USER.**

STEP 5: **IF N IS 0 THEN OUTPUT 1st FIBONACCI NUMBER.**

STEP 6: **IF N IS 1 THEN OUTPUT 2nd FIBONACCI NUMBER.**

STEP 7: **DECLARE A long integer VARIABLE NEXT_NUMBER.**

STEP 8: **INITIALIZE A long integer VARIABLE j=2.**

STEP 9: WHILE J IS LESS THEN N, REPEAT FROM STEP 10 TO STEP 13.

STEP 10: ADD A AND B. ASSIGN SUM OF A AND B TO NEXT_NUMBER.

STEP 11: ASSIGN B's VALUE TO A.

STEP 12: ASSIGN NEXT_NUMBER's VALUE TO B.

STEP 13: INCREMENT J BY 1.

STEP 14: AFTER THE CONDITION IN STEP 9 IS FALSE. OUTPUT THE MOST RECENT VALUE OF B WHICH WILL BE EQUAL TO THE Nth FIBONACCI NUMBER.

STEP 15: STOP.

ALGORITHM # 4

Question 4: Inventory Management

You are tasked with creating an algorithm for a store's inventory management system. Your algorithm should be able to add and remove items from the inventory, update the quantity of existing items, and generate reports of the items and their quantities. Design an algorithm that efficiently manages the store's inventory based on these requirements.

INPUT: STORES INVENTORY (IF IT AREADY EXISTS).

OUTPUT: REPORT OF ITEMS AND THEIR QUANTITIES.

SOLUTION:

STEP 1 : START.

STEP 2: ASK THE USER THAT IF HE ALREADY HAS AN INVENTORY

- **IF YES THEN ASK THE USER TO INPUT THAT INVENTORY.**
- **IF NO THEN INITIALIZE AN EMPTY DATA-STRUCTURE TO STORE THE LIST OF ITEMS**

HAVING ATTRIBUTES LIKE NAME, DESCRIPTION, PRICE AND QUANTITIY.

STEP 3: OUTPUT A MENU TO THE USER SHOWING WHAT DIFFERENT CHOICES HE/SHE CAN MAKE.

STEP 4: INPUT THE USER'S CHOICE.

- IF THE USER WANTS TO ADD ITEMS, GOTO STEP 5.
- IF THE USER WANTS TO REMOVE ITEMS, GOTO STEP 9.
- IF THE USER WANTS TO UPDATE THE QUANTITIY OF AN ITEM THAT ALREADY EXISTS THEN GOTO STEP 13.
- IF THE USER DOEST NOT WANT TO UPDATE ANYTHING THEN GOTO STEP 18.

STEP 5: FOR ADDING ITEMS:

- INPUT NAME, DESCRIPTION, PRICE AND QUANTITIY.

STEP 6: IF THE ITEM ALREADY EXISTS THEN UPDATE THE QUANTITIY BY ADDING IN THE PREVIOUS QUANTITY

STEP 7: IF THE ITEM DOES NOT EXIST THEN CREATE A NEW ENTRY FOR IT IN THE INVENTORY

STEP 8: ASK THE USER IF HE\SHE WANTS TO ADD MORE ITEMS.

- IF YES THEN REPEAT STEP 5 TO STEP 8.
- IF NO THEN GO BACK TO STEP 3.

STEP 9: FOR REMOVING ITEMS:

- INPUT THE NAME OF THE ITEM THAT IS TO BE REMOVED.

STEP 10: IF THE ITEM NAME ENTERED BY THE USER ALREADY EXISTS IN THE INVENTORY, THEN REMOVE IT FROM THE INVENTORY

STEP 11: IF THE ITEM DOES NOT EXIST, THEN OUTPUT AN ERROR MESSAGE.

STEP 12: ASK THE USER IF HE WANTS TO CONTINUE REMOVING ITEMS FROM THE INVENTORY.

- IF YES THEN REAPEAT STEP 9 TO STEP 12.
- IF NO THEN GO BACK TO STEP 3.

STEP 13: FOR UPDATING QUANTITY:

- INPUT THE NAME OF THE ITEM AND THE NEW QUANTITY.

STEP 14: IF THE NAME OF THE ITEM ALREADY EXISTS IN THE INVENTORY THEN UPDATE THE PREVIOUS QUANTITY OF THE ITEM WITH THE NEW QUANTITY.

STEP 15: IF THE ITEM DOES NOT EXIST THEN OUTPUT AN ERROR MESSAGE.

STEP 16: ASK THE USER IF HE\SHE WANTS TO CONTINUE UPDATING THE QUANTITY FOR MORE ITEMS

- IF YES THEN REPEAT STEP 13 TO STEP 16.
- IF NO THEN GO BACK TO STEP 3.

STEP 17: SAVE THE UPDATED DATASTRUCTURE.

STEP 18: GENERATE A REPORT THAT OUTPUTS ALL THE ITEMS NAME AND THEIR QUANTITIES.

STEP 19: STOP.

THE END